UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/826,762	04/16/2004	John Harper	119-0033US	1223

29855 7590 03/30/2007

WONG, CABELLO, LUTSCH, RUTHERFORD & BRUCCULERI, L.L.P.
20333 SH 249
SUITE 600
HOUSTON, TX 77070

EXAMINER	
HSU, JONI	
ART UNIT	**PAPER NUMBER**
2628	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/30/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/826,762	HARPER ET AL.	
	Examiner	Art Unit	
	Joni Hsu	2628	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 December 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-70* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-70* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *5/9/06, 12/15/05*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Amendment

1. In light of Applicant's amendments to the specification and the claims, the objections have been withdrawn.

2. Applicant's arguments filed December 28, 2006 have been fully considered but they are not persuasive.

3. With regard to Claim 75, Applicant argues that Olano (US006717599B1) offers no discussion regarding processing anything on a CPU; quite-the-opposite, Olano focuses on the use of graphics hardware only (page 14).

In reply, the Examiner points out that Olano discloses a general purpose computer having a graphics processor (Col. 3, lines 48-49). The computer (100) comprises the compiler (104) that is implemented so that it transforms high level or shading language statements into statements that are supported by standard graphics hardware (Col. 4, lines 2-4, 35-29). Since the graphics hardware can only operate on statements that are supported by standard graphics hardware, and the compiler is able to transform statements that are not supported by standard graphics hardware, the compiler must inherently not be a part of the graphics hardware. Since the compiler is part of the general purpose computer and is not a part of the graphics hardware, the compiler must inherently be processed by a CPU.

Applicant argues that in addition, Olano's trees are not associated with an image representation by rather with program statements (page 14).

In reply, the Examiner points out that according to the disclosure of this application, a graph associated with an image representation is a graph-like description of an image **task**. The graph-like description may be a node representation of an image ([0008], page 3). Olano teaches that the compiler (104) transforms high level or shading language statements into statements that are supported by standard graphics hardware (Col. 4, lines 35-29). The input to the compiler contains the shading language statements containing nodes (Col. 5, lines 2-13). Therefore, Olano teaches that the program statements are a graph-like description of an image task and is a node representation of an image. Therefore, Olano's trees are associated with an image representation.

Applicant argues that Olano does not propose any of the following concepts: a global domain of definition; a global region of interest; the intersection of the foregoing two items; running routines on a CPU to resolve a node; creating program steps relating to the aforementioned intersection (page 15). Olano does not discuss any division of work between CPU and GPU, or that the claimed program steps are compiled on the CPU and executed on the GPU (page 16).

In reply, the Examiner points out that according to the disclosure of this application, a global domain of definition is a root node domain of definition ([00101], page 27), and the root node is the very highest node before being optimized ([0093], page 23). Olano teaches a domain of definition for a high node (nodes of input 204) before being optimized (*input 204 contains shading language statements, input 204 is converted into a tree data structure representation by compiler 104, the nodes of the tree data*

structure representation are then converted into graphics API statements, reducing the data structure of input 204, Col. 5, lines 5-13, 35-36), and therefore is considered to teach a global domain of definition. Olano teaches marking the nodes of input 204 that are required, and these marked nodes are a region of interest for a high node before being optimized (Col. 5, lines 36-42), and therefore is considered to teach a global region of interest. Olano teaches matching the data structure of input 204 with patterns so as to mark the nodes that are required (Col. 5, lines 32-60), and therefore is considered to teach the intersection of the domain of definition and the region of interest for a high node before being optimized, and therefore is considered to teach the intersection of the foregoing two items. Olano teaches that the routines are run on a compiler (104) to resolve a node (Col. 5, lines 33-60), and the compiler is part of a CPU, as discussed above. Olano teaches creating program steps relating to the aforementioned intersection (Col. 5, lines 56-60). Olano teaches division of work between the compiler and GPU, and that the claimed program steps are compiled on the compiler and executed on the GPU (Col. 5, lines 46-60), and the compiler is part of a CPU, as discussed above.

4. With regard to Claim 79, Applicant argues that McCrossin (US006600840B1) is not an image processing API, but is a "read" and "write" API. McCrossin does not disclose a bundle of API services as claimed; i.e. API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. The McCrossin application does not use an API to call all "filters" involved in the file format conversion (page 17).

In reply, the Examiner points out that McCrossin discloses that an application calls the transformation object using an API call (Col. 7, lines 33-35). Transform object 103 determines an actual image vector 139, which describes the format of the image to be read or written (Col. 6, lines 51-53). Since this describes the format of the image, this is related to image objects and context objects. Filters are accessed by transformation object from filter library 143. Filter library contains a number of different filters available for performing various image transformations (Col. 6, lines 59-62). The pointer to the filter vector in filter object 150 points to filter vector 154, which in the depicted example is a crop filter (Col. 7, lines 19-21). Therefore, McCrossin does teach an image processing API and API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. McCrossin does use an API to call at filters. Claim 79 does not recite the limitation that the filters are involved in the file format conversion.

5. With regard to Claim 84, Applicant argues that McCrossin does not disclose a bundle of API functions as claimed; i.e. API functions to create image objects; API functions to create context objects; API functions to create filter objects; API functions to set filter object parameters; API functions to obtain filter object output; and API functions to convert image objects to context objects (page 17).

In reply, the Examiner points out that McCrossin discloses that an application calls the transformation object using an API call (Col. 7, lines 33-35). Filters are accessed by transformation object from filter library 143. Filter library contains a number of different filters available for performing various image transformations (Col.

6, lines 59-62). According to the disclosure of this application, an image, more commonly, may be created from another image by applying a filter to the existing image ([0057], page 12). Therefore, McCrossin does teach API functions to create image objects; API functions to create filter objects. McCrossin discloses that transform object 103 determines an actual image vector 139, which describes the format of the image to be read or written (Col. 6, lines 51-53). According to the disclosure of this application, creating a context is derived from tools that allow definition of an object in memory ([0055], page 11). Since McCrossin discloses that the actual image vector describes the format of the image to be read or written, this allows definition of an object in memory, and therefore is considered to create context objects, and therefore McCrossin does teach API functions to create context objects. McCrossin discloses that the pointer to the filter environment in filter object 150 points to memory 156, which contains information which may be used by filter vector 154 in converting or transforming image data (Col. 7, lines 21-24), and therefore McCrossin does teach API functions to set filter object parameters. McCrossin discloses that the filters are employed by transformation object 103 to provide the necessary transformation of image data to return image data in a form as specified in image request vector 127 (Col. 6, line 66-Col. 7, line 2), and therefore McCrossin does teach API functions to obtain filter object output; and API functions to convert image objects to context objects.

6. With regard to Claim 1, Applicant argues that Olano does not disclose "a first process requesting a filter from a second process; the related filter and initial image comprising a program." Olano focuses only on the manipulation of program code so that

graphics hardware can perform mathematical derivatives. Olano discusses estimating the size of an anti-aliasing filter; there is no discussion regarding a request for filters between processes (page 18). Olano does not disclose any of the following: which among a plurality of processes compiles the claimed program; or running the claimed compiled program to yield a pixel-image result. Since Olano speaks only to performing mathematical operations, it can not have any relevance to yielding a pixel-image result (pages 19-20).

In reply, the Examiner points out that Olano teaches that application 102 will allow a user to describe images using a high level programming language or shading language, which causes object files residing at the graphics API layer to be executed by the graphics system embodying architecture 100 (Col. 4, lines 15-21). Compiler 104 transforms high level statements to run on a particular graphics system (Col. 4, lines 25-28). Therefore, the application performs a first process requesting from a second process that is performed on the compiler. Derivative values for estimating the size of an anti-aliasing filter are obtained using shading language statements (Col. 6, lines 50-57), and derivative operators are implemented in graphics hardware (Col. 6, lines 21-23). Since the application performing the first process allows a user to describe images using shading language and the compiler performing the second process transforms shading language statements to run on graphics hardware, the first process requests a filter from a second process. The second process compiles the claimed program (Col. 4, lines 25-28). The compiler transfers shading language statements to run on graphics hardware, and therefore the compiled program is run on graphics hardware. One graphics function

applied by graphics hardware is the filter that is applied to the image to yield a pixel-image result (Col. 6, lines 21-23, 50-57).

Applicant argues that McCrossin's transform object "selects" a filter from a library and "employs" that filter; it does not request a filter from another process or deliver a filter in accordance with the request of another process (page 20).

In reply, the Examiner points out that McCrossin is used to teach that the first process defines a relationship between the filter and the initial image (Col. 6, line 46-Col. 7, line 9).

Applicant argues that Olano is a system for doing derivative math in graphics hardware and McCrossin is a way to change file formats. There is no reason to combine references that differ as much as these (page 21).

In reply, the Examiner points out that Olano teaches using an anti-aliasing filter for filtering images (Col. 6, lines 50-55 in Olano) and McCrossin teaches using a number of different filters for filtering images (Col. 6, lines 59-66 in McCrossin), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Olano so that the first process defines a relationship between the filter and the initial image as suggested by McCrossin because McCrossin suggests the advantage of reducing the amount of processing required to convert image data from the original or input format into the desired output format (Col. 1, lines 54-57; Col. 2, lines 3-24 in McCrossin).

7. With regard to Claim 22, Applicant argues that Olano does not imply a first microprocessor running two interrelated processes where one of those processes causes

the creation of a specifically claimed data structure; and then a second microprocessor for running a program created by using the specifically claimed data structure. To the contrary, Olano discusses converting program lines into a node tree and then processing the node tree so that graphics hardware can be used to do derivate math. Olano discuses a frame buffer as illustrative hardware, but it cannot be a buffer for storing a pixel image resulting from running the claimed program. This is because the only "programs" arguing taught by Olano are for doing math, and not for making or editing images. Thus, Olano does not suggest making or editing images using the specifically claimed program (pages 22-23).

In reply, the Examiner points out that the programs for doing math taught by Olano are for doing math for making or editing images (Col. 6, lines 50-55). Therefore, Olano does teach making or editing images using the program.

Applicant argues that McCrossin does not discuss the specifically claimed data structure that comprises the claimed relationship between an initial image and a specific filter (page 23). There is a lack of any disclosure regard a data structure comprising the relationship information (page 24).

In reply, the Examiner disagrees. McCrossin does teach a data structure (API) comprising the relationship between an initial image and a specific filter (*image request vector 127 is received from application 101 and is compared to actual image vector 139 to determine what filters are needed*, Col. 6, line 46-Col. 7, line 9; *application calls the transformation object using API call*, Col. 7, lines 33-35).

8. With regard to Claim 28, Applicant argues that Olano merely proposes making a node tree from program statements and then adapting the tree to perform mathematical derivatives on graphics hardware. Olano doesn't even purport to do so in the context of a second process servicing the request of a first process and Olano doesn't mention optimizing anything (Olano's adaptation of the node tree for math and the graphics processor is not and does not purport to be anything like optimization) (page 25).

In reply, the Examiner points out that the limitation of doing so in the context of a second process servicing the request of a first process is taught by McCrossin (Col. 6, lines 49-58; Col. 6, lines 9-21). Olano discloses optimally transforming high level statements to run on a particular graphics system and reducing the data structure of input 204 (Col. 5, lines 35-36, 56-60), and therefore teaches optimizing a graph representing the result image, as recited in Claim 28.

Applicant argues that McCrossin's transform object selection of vectors for McCrossin's format conversions has nothing to do with creating contexts and rendering to same. McCrossin is focused on the conversion of formats and thus is totally unconcerned with the notion of context creation or use, much less in the specifically claimed manner (page 25).

In reply, the Examiner points out that according to the disclosure of this application, context is a space such as a defined place in memory in which the result of a filtering operation resides ([0045], page 9). McCrossin discloses that the pointer to the filter environment in file object 152 points to memory 162 (Col. 7, lines 29-31). Therefore, McCrossin is considered to teach context creation and use.

9. With regard to Claim 43, Applicant argues that there is no teaching in Olano for an API call having the function to create an image. Olano does not teach associating an image or anything else with a claimed function call. Olano is a system to perform math with graphics hardware, thus Applicant's specifically crafted claims regarding a high-level graphics API are far removed from anything in the Olano disclosure (page 27).

In reply, the Examiner disagrees. Olano teaches a high level graphics API having an API call having the function to create an image (*high level programming language which causes object files residing at the graphics API layer to be executed by the graphics system embodying architecture 100*, Col. 4, lines 18-21).

Applicant argues McCrossin's only APIs are to read and write and thus McCrossin offers nothing in terms of a graphics API, much less defining relationships between such APIs and the specifically claimed objects (page 28).

In reply, the Examiner points out that McCrossin discloses that an application calls the transformation object using an API call (Col. 7, lines 33-35). Filters are accessed by transformation object 103 from filter library 143. Filter library contains a number of different filters available for performing various image transformations (Col. 6, lines 59-62). According to the disclosure of this application, an image, more commonly, may be created from another image by applying a filter to the existing image ([0057], page 12). Therefore, McCrossin does teach graphics API.

10. Applicant argues that there is no motivation to combine Olano, McCrossin and Levy, which are widely ranging references (pages 29-30).

In reply, the Examiner points out that Olano teaches optimizing the program to yield a pixel-image result (Col. 5, lines 35-36, 56-60) and Levy teaches optimizing the program to yield a pixel-image result (*to improve performance the reader application can be designed to cache watermark data to avoid repeated read operations on the same content, cache metadata*, [0184], *window displays metadata, window 106 displays a thumbnail of the image, image attributes (e.g., bits per pixel)*, [0038]), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the step of optimizing includes the step of using a cache look-up to see if the pixel-image result is already in cache as suggested by Levy because Levy suggests the advantage of avoiding repeated read operations on the same content [0184].

11. Applicant argues that there is no motivation to combine Olano, McCrossin and Parikh, which are widely ranging references (page 30).

In reply, the Examiner points out that Olano teaches a compiled program for yielding a pixel-image result (Col. 4, lines 30-35) and Parikh teaches a compiled program for yielding a pixel-image result (Col. 10, lines 1-5), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the compiled program is for a single microprocessor as suggested by Parikh because Parikh suggests the advantage of still being able to perform the graphics processing even when a graphics processor for which the software is written for is not available (Col. 26, lines 29-62).

12. Applicant argues that there is no motivation to combine Olano, McCrossin, Parikh and Doyle, which are four widely ranging references (page 30).

In reply, the Examiner points out that Olano teaches that the CPU performs a process and the GPU performs a process, as discussed above. Doyle teaches that the CPU performs a process and the GPU performs a process (Col. 1, lines 20-26; Col. 2, lines 1-3), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano, McCrossin, and Parikh so that the single microprocessor is a programmable GPU as suggested by Doyle because Doyle suggests the advantage of using the device that would most quickly process the command to process the command (Col. 1, lines 20-26).

13. Applicant argues that there is no motivation to combine Olano, McCrossin and Sturges, which are three widely ranging references (page 31).

In reply, the Examiner points out that Olano teaches a graphics API (Col. 4, lines 18-21) and Sturges teaches a graphics API (Col. 11, lines 42-45), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the first, second and third memories are the same as suggested by Sturges because Sturges suggests the advantage of being able to use unused portions of the frame buffer memory to be employed as system memory (Col. 1, lines 35-41).

14. Applicant argues that there is no motivation to combine Olano, McCrossin and Stokes, which are three widely ranging references (page 31).

In reply, the Examiner points out that Olano teaches that the second process comprises a suite of graphics services functions (Col. 4, lines 15-21) and Stokes teaches a graphics service function of converting color schemes (Col. 5, lines 63-67), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the second process inserts nodes into the graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme as suggested by Stokes because Stokes suggests that this is needed because the color data generated by an image-capturing device are generally in a device-specific color space that is different from the color space or spaces used by the image-processing application for image editing (Col. 1, lines 40-44).

15. Applicant argues that there is no motivation to combine Olano and French which are two widely ranging references (page 32).

In reply, the Examiner points out that Olano teaches nodes of a graph associated with an image representation (Col. 5, lines 10-13) and French teaches nodes of a graph associated with an image representation (Col. 3, lines 51-54), and therefore the references are related. It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Olano so that the first node is a root node as suggested by French. French suggests that many elements of media context are heavily time dependent (Col. 3, lines 10-28), and root nodes are needed in order to

integrate a time context and time inheritance into a graph oriented scene modeling system without requiring the adoption or learning of a new programming language (Col. 4, lines 32-36; Col. 6, lines 15-23).

Claim Rejections - 35 USC § 102

16. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

17. Claim 75 is rejected under 35 U.S.C. 102(e) as being anticipated by Olano (US006717599B1).

Olano discloses a general purpose computer having a graphics processor (Col. 3, lines 48-49). The computer (100) comprises the compiler (104) that is implemented so that it transforms high level or shading language statements into statements that are supported by standard graphics hardware (Col. 4, lines 2-4, 35-29). Since the graphics hardware can only operate on statements that are supported by standard graphics hardware, and the compiler is able to transform statements that are not supported by standard graphics hardware, the compiler must inherently not be a part of the graphics hardware. Since the compiler is part of the general purpose computer and is not a part of

the graphics hardware, the compiler must inherently be processed by a CPU. According to the disclosure of this application, a graph associated with an image representation is a graph-like description of an image **task**. The graph-like description may be a node representation of an image ([0008], page 3). Olano teaches that the compiler (104) transforms high level or shading language statements into statements that are supported by standard graphics hardware (Col. 4, lines 35-29). The input to the compiler contains the shading language statements containing nodes (Col. 5, lines 2-13). Therefore, Olano teaches that the program statements are a graph-like description of an image task and is a node representation of an image. Therefore, Olano's trees are associated with an image representation. According to the disclosure of this application, a global domain of definition is a root node domain of definition ([00101], page 27), and the root node is the very highest node before being optimized ([0093], page 23). Olano teaches a domain of definition for a high node (nodes of input 204) before being optimized (*input 204 contains shading language statements, input 204 is converted into a tree data structure representation by compiler 104, the nodes of the tree data structure representation are then converted into graphics API statements, reducing the data structure of input 204*, Col. 5, lines 5-13, 35-36), and therefore is considered to teach a global domain of definition. Olano teaches marking the nodes of input 204 that are required, and these marked nodes are a region of interest for a high node before being optimized (Col. 5, lines 36-42), and therefore is considered to teach a global region of interest. Olano teaches matching the data structure of input 204 with patterns so as to mark the nodes that are required (Col. 5, lines 32-60), and therefore is considered to teach the intersection of the domain of definition and the region of interest for a high node before being

optimized, and therefore is considered to teach the intersection of the foregoing two items. Olano teaches that the routines are run on a compiler (104) to resolve a node (Col. 5, lines 33-60), and the compiler is part of a CPU, as discussed above. Olano teaches creating program steps relating to the aforementioned intersection (Col. 5, lines 56-60). Olano teaches division of work between the compiler and GPU, and that the claimed program steps are compiled on the compiler and executed on the GPU (Col. 5, lines 46-60), and the compiler is part of a CPU, as discussed above. Therefore, Olano describes a method for converting a first image representation into a second image representation, comprising the steps of: creating a first graph associated with the first image representation where software routines for creating such graph execute on a CPU (Col. 5, lines 3-10; Col. 4, lines 30-35; Col. 3, lines 48-53), determining an intersection of the first graph's global domain of definition and global region of interest; resolving a first node in the first graph by running software routines on the CPU to (i) determine if the first node may be combined with a second node and (ii) create program steps for calculating and storing only portions of any intermediary image that relate to the intersection (Col. 5, lines 33-60), the program steps for compilation on the CPU and execution on a GPU (Col. 4, lines 30-35; Col. 3, lines 48-53).

Thus, it reasonably appears that Olano describes or discloses every element of the claim, and therefore anticipates the claim subject.

18. Claims 79-84 are rejected under 35 U.S.C. 102(e) as being anticipated by McCrossin (US006600840B1).

19. With regard to Claim 79, McCrossin discloses that an application calls the transformation object using an API call (Col. 7, lines 33-35). Transform object 103 determines an actual image vector 139, which describes the format of the image to be read or written (Col. 6, lines 51-53). Since this describes the format of the image, this is related to image objects and context objects. Filters are accessed by transformation object from filter library 143. Filter library contains a number of different filters available for performing various image transformations (Col. 6, lines 59-62). The pointer to the filter vector in filter object 150 points to filter vector 154, which in the depicted example is a crop filter (Col. 7, lines 19-21). Therefore, McCrossin does teach an image processing API and API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. McCrossin does use an API to call at filters. Therefore, McCrossin describes an image processing application program interface embodied on one or more computer readable media (Col. 6, lines 25-30), comprising a first group of services related to filter objects (Col. 6, lines 59-67); a second group of service related to image objects; a third group of services related to context objects (Col. 6, lines 51-53); and a fourth group of services related to vector objects (Col. 7, lines 14-17).

20. With regard to Claim 80, McCrossin describes that the first group of services comprise first functions to create filter objects; second functions to set filter object parameters; and third functions to obtain filter object output (Col. 2, lines 25-32).

21. With regard to Claim 81, McCrossin describes that the second group of services comprise first function to create image objects (Col. 6, lines 51-56); and second functions to render an image object to a context object (Col. 9, lines 7-18).

22. With regard to Claim 82, McCrossin describes that the third group of services comprise first functions to create context objects (Col. 5, lines 46-67).

23. With regard to Claim 83, McCrossin describes that the fourth group of services comprise first functions to create vector objects (Col. 7, lines 15-32).

24. With regard to Claim 84, McCrossin discloses that an application calls the transformation object using an API call (Col. 7, lines 33-35). Filters are accessed by transformation object from filter library 143. Filter library contains a number of different filters available for performing various image transformations (Col. 6, lines 59-62). According to the disclosure of this application, an image, more commonly, may be created from another image by applying a filter to the existing image ([0057], page 12). Therefore, McCrossin does teach API functions to create image objects; API functions to create filter objects. McCrossin discloses that transform object 103 determines an actual image vector 139, which describes the format of the image to be read or written (Col. 6, lines 51-53). According to the disclosure of this application, creating a context is derived from tools that allow definition of an object in memory ([0055], page 11). Since McCrossin discloses that the actual image vector describes the format of the image to be read or written, this allows definition of an object in memory, and therefore is considered

to create context objects, and therefore McCrossin does teach API functions to create context objects. McCrossin discloses that the pointer to the filter environment in filter object 150 points to memory 156, which contains information which may be used by filter vector 154 in converting or transforming image data (Col. 7, lines 21-24), and therefore McCrossin does teach API functions to set filter object parameters. McCrossin discloses that the filters are employed by transformation object 103 to provide the necessary transformation of image data to return image data in a form as specified in image request vector 127 (Col. 6, line 66-Col. 7, line 2), and therefore McCrossin does teach API functions to obtain filter object output; and API functions to convert image objects to context objects. Therefore, McCrossin discloses an application program interface for facilitating image processing (Col. 6, lines 25-30), the application program interface comprising functions to create image objects; create context objects (Col. 6, lines 51-53); create filter objects; set filter object parameters; obtain filter object output (Col. 2, lines 25-32); and convert image objects into context objects (Col. 6, lines 51-53).

25. Thus, it reasonably appears that McCrossin describes or discloses every element of Claims 78-84 and therefore anticipates the claims subject.

Claim Rejections - 35 USC § 103

26. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject

matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

27. The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

28. Claims 1, 2, 4-6, 8, 11-23, 25-32, 36-41, 43-47, 51-56, 58, 61-64, 66-74, and 85 are rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) in view of McCrossin (US006600840B1).

29. With regard to Claim 1, Olano teaches that application 102 will allow a user to describe images using a high level programming language or shading language, which causes object files residing at the graphics API layer to be executed by the graphics system embodying architecture 100 (Col. 4, lines 15-21). Compiler 104 transforms high level statements to run on a particular graphics system (Col. 4, lines 25-28). Therefore, the application performs a first process requesting from a second process that is performed on the compiler. Derivative values for estimating the size of an anti-aliasing filter are obtained using shading language statements (Col. 6, lines 50-57), and derivative operators are implemented in graphics hardware (Col. 6, lines 21-23). Since the

application performing the first process allows a user to describe images using shading language and the compiler performing the second process transforms shading language statements to run on graphics hardware, the first process requests a filter from a second process. The second process compiles the claimed program (Col. 4, lines 25-28). The compiler transfers shading language statements to run on graphics hardware, and therefore the compiled program is run on graphics hardware. One graphics function applied by graphics hardware is the filter that is applied to the image to yield a pixel-image result (Col. 6, lines 21-23, 50-57). Therefore, Olano discloses a method of editing an initial image, comprising the steps of a first process requesting a filter from a second process; the related filter and initial image comprising a program (Col. 6, line 50-Col. 7, line 3), said second process compiling the program, yielding a compiled program; running at least a portion of the compiled program to apply a function of the filter to the image, yielding an pixel-image result (Col. 2, lines 52-56).

However, Olano does not teach that the first process defines a relationship between the filter and the initial image. However, McCrossin discloses a method of editing an initial image, comprising the steps of a first process requesting a filter from a second process; said first process defining a relationship between the filter and the initial image (Col. 6, line 46-Col. 7, line 9).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Olano so that the first process defines a relationship between the filter and the initial image as suggested by McCrossin because McCrossin suggests the advantage of reducing the amount of processing required to convert image data from the original or input format into the desired output format (Col.

1, lines 54-57; Col. 2, lines 3-24). Olano teaches using an anti-aliasing filter for filtering images (Col. 6, lines 50-55 in Olano) and McCrossin teaches using a number of different filters for filtering images (Col. 6, lines 59-66 in McCrossin), and therefore the references are related.

30. With regard to Claim 2, Olano describes having the additional step of optimizing the program (Col. 4, lines 30-36).

31. With regard to Claim 4, Olano describes that the step of optimizing includes the step of calculating an intersection, the intersection representing an area where the pixel-image result is both defined and in a result region requested of the second process (Col. 5, lines 33-60).

32. With regard to Claim 5, Olano describes the step of using the calculated intersection to limit the number of pixels that require calculation during the running of the compiled program (Col. 5, lines 33-60).

33. With regard to Claim 6, Olano describes the step of using the calculated intersection to limit the amount of memory necessary for storing calculated images (Col. 5, lines 33-60).

34. With regard to Claim 8, Olano describes that the compiled program comprises a component for a first microprocessor and a component for a second microprocessor (Col. 3, lines 45-53).

35. With regard to Claim 11, Olano describes that the first microprocessor is a CPU and the second microprocessor is a GPU (Col. 3, lines 45-53).

36. With regard to Claim 12, Olano describes that the first and second microprocessors are both CPUs (Col. 11, lines 49-50).

37. With regard to Claim 13, Olano describes that the first and second microprocessors are both GPUs (Col. 11, lines 53-55).

38. With regard to Claim 14, Olano describes that the initial image is only a color (Col. 1, lines 38-42).

39. With regard to Claim 15, Olano describes that the first process is an application program (Col. 4, lines 12-21).

40. With regard to Claim 16, Olano describes that the second process comprises a suite of graphics services functions (Col. 4, lines 15-21).

41. With regard to Claim 17, Olano describes that an operating system comprises the second process (Col. 4, lines 48-53).

42. With regard to Claim 18, Olano describes that the first process requests a high-level filter and the second process responds with an object representing a lower-level filter (Col. 4, lines 30-36).

43. With regard to Claim 19, Olano describes that the first process (102, Figure 1) and second process (104) run on a CPU (904, Figure 9) and the compiled program runs on a GPU (903) (Col. 11, lines 53-58; Col. 2, lines 53-56; Col. 4, lines 11-21, 30-36).

44. With regard to Claims 20 and 21, these claims are both similar in scope to Claim 19, and therefore are rejected under the same rationale.

45. With regard to Claim 22, Olano describes a system for editing an initial image (Col. 6, lines 50-67), comprising a first microprocessor running a first process and a second process for servicing requests from the first process (Col. 3, lines 45-53; Col. 4, lines 11-35); a memory (802, Figure 8) for storing a filter (Col. 10, lines 20-28), the filter requested by the first process (Col. 4, lines 11-21); a second memory (802) for storing a data structure comprising information used in the first process (Col. 10, lines 20-28; Col. 6, lines 50-67), the first process causing the creation of the data structure (Col. 2, lines 45-52); a second microprocessor for running a program created using the data structure (Col. 4, lines 30-35, 54-60); a third memory (818) for storing a pixel image resulting

from running the program (Col. 11, lines 14-18). The programs for doing math taught by Olano are for doing math for making or editing images (Col. 6, lines 50-55). Therefore, Olano teaches making or editing images using the program.

However, Olano does not teach that the data structure comprises a relationship between the initial image and the filter. However, McCrossin discloses a data structure (API) comprising the relationship between an initial image and a specific filter (*image request vector 127 is received from application 101 and is compared to actual image vector 139 to determine what filters are needed*, Col. 6, line 46-Col. 7, line 9; *application calls the transformation object using API call*, Col. 7, lines 33-35). This would be obvious for the same reasons given in the rejection for Claim 1.

46. With regard to Claim 23, Olano describes that the first and second memories are the same (802, Figure 8; Col. 10, lines 20-28; Col. 6, lines 50-67).

47. With regard to Claim 25, Olano describes that the first and second memories are in system memory (802, Figure 8) and the third memory is in video memory (818) (Col. 10, lines 20-28; Col. 11, lines 14-18).

48. With regard to Claim 26, Olano describes that the first microprocessor processes the data structure to produce the program for use on the second microprocessor (Col. 3, lines 45-53; Col. 4, lines 30-35).

49. With regard to Claim 27, Olano describes that the second microprocessor, under control of the program, causes the pixel image to be stored in the third memory (818) (Col. 11, lines 21-23, 32-33; Col. 4, lines 30-36).

50. With regard to Claim 28, Olano describes a method of creating a result image comprising the steps of a first process requesting the creation of a result image (Col. 4, lines 11-21); a second process servicing the requests of the first process, the servicing comprising the steps of optimizing a graph representing the result image; compiling the optimized graph; causing the rendering of the compiled optimized graph (Col. 2, lines 45-56; Col. 4, lines 30-35). Olano discloses optimally transforming high level statements to run on a particular graphics system and reducing the data structure of input 204 (Col. 5, lines 35-36, 56-60), and therefore teaches optimizing a graph representing the result image.

However, Olano does not teach that the first process requests the creation of a context; the first process indicating parameters associated with the creation of the result image; the first process requesting that the result image be rendered to the context. However, McCrossin discloses that the first process requests the creation of a context (Col. 5, lines 49-50); the first process indicating parameters associated with the creation of the result image; the first process requesting that the result image be rendered to the context (Col. 6, lines 51-58; Col. 6, lines 9-21). According to the disclosure of this application, context is a space such as a defined place in memory in which the result of a filtering operation resides ([0045], page 9). McCrossin discloses that the pointer to the filter environment in file object 152 points to memory 162 (Col. 7, lines 29-31).

Therefore, McCrossin is considered to teach context creation and use. This would be obvious for the same reasons given in the rejection for Claim 1.

51. With regard to Claim 29, Olano describes that the creation of the result image comprises editing a pre-existing image (Col. 6, lines 50-67).

52. With regard to Claim 30, Claim 30 is similar in scope to Claim 4, and therefore is rejected under the same rationale.

53. With regard to Claim 31, Olano describes that the step of optimizing a graph representing the first image (Col. 4, lines 30-36) comprises the step of analyzing adjacent nodes in the graph for the purpose of attempting to consolidate nodes (Col. 5, lines 33-45).

54. With regard to Claim 32, Claim 32 is similar in scope to Claim 31, and therefore is rejected under the same rationale.

55. With regard to Claim 36, Olano describes that the first process requests the output of a graph programmatically assembled by the first process (Col. 4, lines 11-35; Col. 2, lines 45-56).

However, Olano does not teach that the graph comprises one or more pre-defined filters. However, McCrossin discloses that the graph comprises one or more pre-defined

filters (Col. 6, lines 59-67). This would be obvious for the same reasons given in the rejection for Claim 1.

56. With regard to Claim 37, Olano describes that the first process programmatically assembled the graph in cooperation with the second process (Col. 4, line 61-Col. 5, line 19).

57. With regard to Claims 38-40, these claims are similar in scope to Claim 15-17 respectively, and therefore are rejected under the same rationale. With regard to Claim 41, Claim 41 is similar in scope to Claim 40, and therefore is rejected under the same rationale.

58. With regard to Claim 43, Claim 43 is similar in scope to Claim 28, except Claim 43 has the additional limitation that the first process and second process are running on a first microprocessor, and the rendering occurs on a second microprocessor. Olano discloses that the first process and second process are running on a first microprocessor, and the rendering occurs on a second microprocessor (Col. 3, lines 48-53; Col. 4, lines 30-35). Olano teaches a high level graphics API having an API call having the function to create an image (*high level programming language which causes object files residing at the graphics API layer to be executed by the graphics system embodying architecture 100*, Col. 4, lines 18-21). McCrossin discloses that an application calls the transformation object using an API call (Col. 7, lines 33-35). Filters are accessed by transformation object 103 from filter library 143. Filter library contains a number of

different filters available for performing various image transformations (Col. 6, lines 59-62). According to the disclosure of this application, an image, more commonly, may be created from another image by applying a filter to the existing image ([0057], page 12). Therefore, McCrossin teaches graphics API. Therefore, Claim 43 is rejected under the same rationale as Claim 28.

59. With regard to Claims 44-47 and 51-56, these claims are similar in scope to Claims 29-32 and 36-41 respectively, and therefore are rejected under the same rationale.

60. With regard to Claim 58, Olano describes a method for providing a high level interface to a graphics processing resource (Col. 4, lines 47-53) comprising a first process requesting performance of a task through one or more function calls serviced by the graphics processing resource (Col. 4, lines 11-21), the function calls selected from one or more of the following options: (i) creating of an image (Col. 10, lines 59-65); said request having an object associated therewith, the object comprising one of the following: an image (Col. 4, lines 15-21); the graphics processing resource servicing the request (Col. 4, lines 30-35).

However, Olano does not teach that the request defines a relationship between at least one of the functions and one of the objects. However, McCrossin discloses that the request defines a relationship between at least one of the functions and one of the objects (Col. 6, line 46-Col. 7, line 9). This would be obvious for the same reasons given in the rejection for Claim 1.

61. With regard to Claim 61, Olano describes the additional step of creating a graph representing an image (Col. 5, lines 3-19).

62. With regard to Claim 62, Olano describes the additional step of optimizing the graph (Col. 4, lines 30-35; Col. 5, lines 3-19).

63. With regard to Claims 63, 64, 66, and 67, these claims are similar in scope to Claims 58, 59, 61, and 62 respectively, and therefore are rejected under the same rationale.

64. With regard to Claim 68, Olano does not teach that the function calls include the option of creating a filter. However, McCrossin discloses that the function calls include the option of creating a filter (Col. 6, lines 59-67). This would be obvious for the same reasons given in the rejection for Claim 1.

65. With regard to Claim 69, Olano does not teach that the function calls include the option of setting arguments associated with the filter. However, McCrossin discloses that the function calls include the option of setting arguments associated with the filter (Col. 7, lines 1-7; Col. 6, lines 9-21). This would be obvious for the same reasons given in the rejection for Claim 1.

66. With regard to Claim 70, Claim 70 is similar in scope to Claim 69, and therefore is rejected under the same rationale.

67. With regard to Claim 71, Olano does not teach that said another object associated with the request may comprise a filter. However, McCrossin discloses that said another object associated with the request may comprise a filter (Col. 6, lines 59-67). This would be obvious for the same reasons given in the rejection for Claim 1.

68. With regard to Claim 72, Olano does not teach that another object associated with the request may be a vector. However, McCrossin discloses that another object associated with the request may be a vector (Col. 7, lines 14-17). This would be obvious for the same reasons given in the rejection for Claim 1.

69. With regard to Claim 73, Claim 73 is similar in scope to Claim 72, and therefore is rejected under the same rationale.

70. With regard to Claim 74, Olano describes a method for rendering an image comprising the steps of: a first process running on a CPU requesting the creation of an image; a graphics services resource, responding to the request by running a first routine on the CPU (Col. 4, lines 30-36; Col. 3, lines 48-53), the first routine for optimizing a graph representing the image; the graphics services resource causing the rendering of the graph representing the image, the rendering occurring on a GPU (Col. 4, lines 30-36).

However, Olano does not teach that the first process requests the rendering of the image to a specified context. However, McCrossin discloses that the first process

requests the rendering of the image to a specified context (Col. 6, lines 9-16). This would be obvious for the same reasons given in the rejection for Claim 1.

71. With regard to Claim 85, Olano discloses a computer-readable medium having computer executable instructions for performing the method (Col. 12, lines 58-67) recited in Claim 1.

72. Claims 3, 33-35, and 48-50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) and McCrossin (US006600840B1) in view of Levy (US 20020033844A1).

73. With regard to Claim 3, Olano and McCrossin are relied upon for the teachings as discussed above relative to Claim 2.

However, Olano and McCrossin do not teach that the step of optimizing includes the step of using a cache look-up to see if the pixel-image result is already in cache. However, Levy describes that the step of optimizing includes the step of using a cache look-up to see if the pixel-image result is already in cache [0184, 0021, 0038].

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the step of optimizing includes the step of using a cache look-up to see if the pixel-image result is already in cache as suggested by Levy because Levy suggests the advantage of avoiding repeated read operations on the same content [0184]. Olano teaches optimizing the program to yield a pixel-image result (Col. 5, lines 35-36, 56-60) and Levy teaches

optimizing the program to yield a pixel-image result (*to improve performance the reader application can be designed to cache watermark data to avoid repeated read operations on the same content, cache metadata*, [0184], *window displays metadata, window 106 displays a thumbnail of the image, image attributes (e.g., bits per pixel)*, [0038]), and therefore the references are related.

74. With regard to Claim 33, Olano discloses the step of analyzing adjacent nodes (Col. 5, lines 3-19, 33-45) and storing the result of such analysis in memory (Col. 5, lines 61-67).

However, Olano does not teach the step of checking a cache to determine if the result of such analysis is available in a memory. However, Levy describes the step of checking a cache to determine if the pixel-image result is available in a memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

75. With regard to Claim 34, Olano discloses the step of optimizing the graph (Col. 4, lines 30-36; Col. 5, lines 3-19) and storing the optimized graph in memory (Col 5, lines 3-19, 33-45).

However, Olano does not teach the step of checking a cache to determine if the graph has already been optimized. However, Levy describes the step of checking a cache to determine if the pixel-image result is already in memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

76. With regard to Claim 35, Olano discloses the step of optimizing the graph (Col. 4, lines 30-36; Col. 5, lines 3-19) and storing the result of rendering the graph in memory (Col. 21-23, 32-33).

Olano does not teach the step of checking a cache to determine if the result of rendering the graph is available in a memory. However, Levy describes the step of checking a cache to determine if the pixel-image result is available in a memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

77. With regard to Claims 48-50, these claims are similar in scope to Claims 33-35 respectively, and therefore are rejected under the same rationale.

78. Claims 7, 9, 60, and 65 are rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) and McCrossin (US006600840B1) in view of Parikh (US006411301B1).

79. With regard to Claim 7, Olano and McCrossin are relied upon for the teachings as discussed above relative to Claim 1.

However, Olano and McCrossin do not teach that the compiled program is for a single microprocessor. However, Parikh describes that the compiled program is for a single microprocessor (Col. 26, lines 29-62).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the

compiled program is for a single microprocessor as suggested by Parikh because Parikh suggests the advantage of still being able to perform the graphics processing even when a graphics processor for which the software is written for is not available (Col. 26, lines 29-62). Olano teaches a compiled program for yielding a pixel-image result (Col. 4, lines 30-35) and Parikh teaches a compiled program for yielding a pixel-image result (Col. 10, lines 1-5), and therefore the references are related.

80. With regard to Claim 9, Olano does not teach that the single microprocessor is a CPU. However, Parikh describes that the single microprocessor is a CPU (Col. 26, lines 29-62). This would be obvious for the same reasons given in the rejection for Claim 7.

81. With regard to Claim 60, Olano does not teach that the request is serviced using an emulator to run a GPU program on a CPU. However, Parikh describes that the request is serviced using an emulator to run a GPU program on a CPU (Col. 26, lines 29-62). This would be obvious for the same reasons given in the rejection for Claim 7.

82. With regard to Claim 65, Claim 65 is similar in scope to Claim 60, and therefore is rejected under the same rationale.

83. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1), McCrossin (US006600840B1), and Parikh (US006411301B1) in view of Doyle (US006867779B1).

Olano, McCrossin, and Parikh are relied upon for the teachings as discussed above relative to Claim 7.

However, Olano, McCrossin, and Parikh do not teach that the single microprocessor is a programmable GPU. However, Doyle discloses dividing up the rendering between the CPU and the GPU based on the progress of one device of the two devices (Col. 1, lines 20-26; Col. 2, lines 1-3). Therefore, if the CPU is busy, the rendering is only performed in the GPU. Therefore, Doyle discloses that the single microprocessor is a programmable GPU.

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano, McCrossin, and Parikh so that the single microprocessor is a programmable GPU as suggested by Doyle because Doyle suggests the advantage of using the device that would most quickly process the command to process the command (Col. 1, lines 20-26). Olano teaches that the CPU performs a process and the GPU performs a process, as discussed above. Doyle teaches that the CPU performs a process and the GPU performs a process (Col. 1, lines 20-26; Col. 2, lines 1-3), and therefore the references are related.

84. Claim 24 is rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) and McCrossin (US006600840B1) in view of Sturges (US005854637A).

Olano and McCrossin are relied upon for the teachings as discussed above relative to Claim 22.

However, Olano and McCrossin do not teach that the first, second and third memories are the same. However, Sturges discloses a system with an API (Col. 11, lines 42-45), comprising using a shared memory for both the system memory and the frame buffer (Col. 3, lines 7-15).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the first, second and third memories are the same as suggested by Sturges because Sturges suggests the advantage of being able to use unused portions of the frame buffer memory to be employed as system memory (Col. 1, lines 35-41). Olano teaches a graphics API (Col. 4, lines 18-21) and Sturges teaches a graphics API (Col. 11, lines 42-45), and therefore the references are related.

85. Claims 42 and 57 are rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) and McCrossin (US006600840B1) in view of Stokes (US006977661B1).

86. With regard to Claim 42, Olano and McCrossin are relied upon for the teachings as discussed above relative to Claim 28. Olano teaches that the second process inserts nodes into the graph for performing certain functions (Col. 5, lines 3-19).

However, Olano and McCrossin do not teach that the functions include functions for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme. However, Stokes discloses

converting an original color scheme to an operating color scheme and converting the operating color scheme back to the original color scheme (Col. 5, lines 63-67).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the second process inserts nodes into the graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme as suggested by Stokes because Stokes suggests that this is needed because the color data generated by an image-capturing device are generally in a device-specific color space that is different from the color space or spaces used by the image-processing application for image editing (Col. 1, lines 40-44). Olano teaches that the second process comprises a suite of graphics services functions (Col. 4, lines 15-21) and Stokes teaches a graphics service function of converting color schemes (Col. 5, lines 63-67), and therefore the references are related.

87. With regard to Claim 57, Claim 57 is similar in scope to Claim 42, and therefore is rejected under the same rationale.

88. Claim 59 is rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) and McCrossin (US006600840B1) in view of Doyle (US006867779B1).

Olano and McCrossin are relied upon for the teachings as discussed above relative to Claim 58.

However, Olano and McCrossin do not teach that the request is serviced using a GPU and a CPU. However, Doyle teaches that the request is serviced using a GPU and a CPU (Col. 2, lines 1-3).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the devices of Olano and McCrossin so that the request is serviced using a GPU and a CPU as suggested by Doyle because Doyle suggests that this accelerates the speed by which a three-dimensional image can be rendered (Col. 1, lines 7-10).

89. Claims 76 and 77 are rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) in view of Levy (US 20020033844A1).

90. With regard to Claim 76, Olano is relied upon for the teachings as discussed above relative to Claim 75. Olano teaches the step of determining if the first node may be combined with the second node, and the result is stored in memory (Col. 5, lines 33-60).

However, Olano does not teach the step of checking a cache to determine if there is a result is already in memory regarding such determination regarding combining nodes. However, Levy discloses the step of checking a cache to determine if the pixel-image result is already in memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

91. With regard to Claim 77, Olano discloses the step of resolving the first node and storing the resolution of the first node in memory (Col. 5, lines 33-60).

Olano does not teach that the step of resolving the first node comprises the step of checking a cache to determine if there is a result in memory regarding the resolution of the first node. However, Levy discloses the step of checking a cache to determine if the pixel-image result is already in memory [0184, 0021, 0038]. This would be obvious for the same reasons given in the rejection for Claim 3.

92. Claim 78 is rejected under 35 U.S.C. 103(a) as being unpatentable over Olano (US006717599B1) in view of French (US006266053B1).

Olano is relied upon for the teachings as discussed above relative to Claim 75.

However, Olano does not teach that the first node is a root node. However, French describes that the first node is a root node (Col. 3, lines 65-66).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify the device of Olano so that the first node is a root node as suggested by French. French suggests that many elements of media context are heavily time dependent (Col. 3, lines 10-28), and root nodes are needed in order to integrate a time context and time inheritance into a graph oriented scene modeling system without requiring the adoption or learning of a new programming language (Col. 4, lines 32-36; Col. 6, lines 15-23). Olano teaches nodes of a graph associated with an image representation (Col. 5, lines 10-13) and French teaches nodes of a graph associated with an image representation (Col. 3, lines 51-54), and therefore the references are related.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Joni Hsu whose telephone number is 571-272-7785. The examiner can normally be reached on M-F 8am-5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ulka Chauhan can be reached on 571-272-7782. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

JH

KEE M. TUNG
SUPERVISORY PATENT EXAMINER